НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: Система управління програмованими пристроями взаємодії з користувачем

Виконав: студент IV курсу, групи __КВ-51__
(шифр групи)

_____Базильський Лев Олександрович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ___доц.каф.СПСКС, к.т.н. Петрашенко А.В _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
Базильського Лева Олександровича
(прізвище, ім’я, по батькові)

1. Тема проекту Система управління програмованими пристроями взаємодії з користувачем,
керівник проекту доц.каф. СПСКС, к.т.н. Петрашенко А.В.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту див. Технічне завдання

4. Зміст пояснювальної записки

- Аналіз існуючих рішень та обґрунтування теми дипломного проекту
- Структура програмних засобів
- Опис розроблених алгоритмів
- Аналіз розробленої системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Взаємодія зовнішніх пристроїв та ПК за допомогою USB інтрефейсу. Схема структурна
- Інтерфейс USB HID бібліотеки. Схема алгоритму
- Монітор HID пристроїв. Блок-схема алгоритму
- Структура кросплатформенної HID бібліотеки. Діаграми класів.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Кдятченко Я.М. доцент		

7. Дата видачі завдання «___» _____2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	15.04.2019	
2.	Розроблення та узгодження технічного завдання	30.04.2019	
3.	Аналіз існуючих рішень	05.05.2019	
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019	
6.	Підготовка графічної частини дипломного проекту	20.05.2019	
7.	Оформлення документації дипломного проекту	25.05.2019	
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019	

Студент _____ Базильський Л.О.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Петрашенко А.В.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (50 с., 13 рис., 4 додатки)

Об’єкт розробки – створення комп’ютерної системи управління програмованими пристроями взаємодії з користувачем.

Система управління дозволяє: здійснювати моніторинг підключених пристроїв; зчитувати дані з пристрою та відправляти їх йому, обробляти різноманітні події від пристрою, блокувати їх та відтворювати інші. Система написана з використанням мови програмування C++ та фреймворку Qt.

В ході розробки:

- проведено аналіз USB HID протоколу для взаємодії з користувацькими пристроями;
- сформульовані вимоги до системи взаємодії з пристроями;
- розроблено бібліотеку для взаємодії з користувацькими пристроями;
- розроблено користувацький додаток для управління і моніторингу роботи пристроїв;

Ця система може бути легко доповнена або стати основою для створення власного додатку для взаємодії з пристроями, що працюють за допомогою USB HID протоколу та які не були наведені в даному дипломному проекті.

Ключові слова:

СИСТЕМА УПРАВЛІННЯ ПРИСТРОЯМИ, USB HID ПРОТОКОЛ, МОВА ПРОГРАМУВАННЯ C++, QT ФРЕЙМВОРК.

Abstract

Qualification work includes explanatory note (50 pages, 13 images, 4 attachments)

Subject of developments is creation of computer control system that handles programmable HID devices.

Control system allows: to monitor connected devices; read and write data to connected devices, handling different device events, to block and repeat other. System is written in C++ using Qt framework.

During development process:

- USB HID protocol was analyzed;
- system requirements were articulated;
- HID devices handling library was developed;
- user application to manage and monitor connected devices was developed;

This system could easily be updated or be used as base for extended application that handles device interactions with devices that works via USB HID protocol and which weren't covered in this project.

Ключові слова:

DEVICE CONTROL SYSTEM, USB HID PROTOCOL, C++ PROGRAMMING LANGUAGE, QT FRAMEWORK.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Система управління	4		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Система управління	2		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Система управління	50		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045430.001 ОА		
Розробив		Базильський ЛО			Система управління програмованими пристроями взаємодії з користувачем Опис альбому	Літ.	Аркуш 1 / Аркушів 3
Перевірив		ПетрашенкоА.В					
Консульт.						НТУУ «КПІ ім. І.Сікорського» Кафедра СПіСКС Група КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.005 Д1	Система управління	1		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Взаємодія зовнішніх			
			пристроїв та ПК за			
			допомогою USB інтерфейс			
	А4	ІАЛЦ.045490.006 Д2	Система управління	1		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Інтерфейс USB HID			
			бібліотеки			
	А4	ІАЛЦ.045490.007 Д3	Система управління	1		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Монітор HID пристроїв			
	А4	ІАЛЦ.045490.008 Д4	Система управління	1		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Структура			
			кросплатфоменної HID			
			бібліотеки			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмне забезпечення роботи з системою 1С на мобільних пристроях».

Галузь застосування: Організація управління користувацькими переферійними пристроями.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання дипломного проекту першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення користувацького додатку для управління програмованими пристроями, з якими безпосередньо взаємодіють користувачі під час роботи з персональним комп'ютером.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь якою операційною системою (Windows, Linux, MacOS);
- наявність моніторингу підключених пристроїв;
- можливість відправляти та отримувати дані на пристрій;
- можливість отримувати основну інформацію про пристрій; ивості.

5.2. Вимоги до апаратного забезпечення

- Процесор з тактовою частотою: 2 ГГц;
- Оперативна пам’ять: 2 Гб;

5.3. Вимоги програмного забезпечення

- Операційна система Windows, Linux, MacOS;
- Встановлений фреймворк Qt.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.004 ПЗ	Система управління	50		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Пояснювальна записка.			
				1		
	А4	ІАЛЦ.045490.005 Д1	Система управління			
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Схема алгоритму	1		
	А4	ІАЛЦ.045490.006 Д2	Система управління			
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Схема алгоритму	1		

		№ докум.	Підпис		ІАЛЦ.045490.003 ТП
Змін.	Арк.			Дата	
Розробив		Базильський Л.О.			Система управління програмованими пристроями взаємодії з користувачем. Відомість технічного проекту
Перевірив		Петрашенко А.В.			
Н. контроль		Клятченко Я.М.			
Затвердив		Тарасенко В.П.			

Літ.	Аркуш	Аркушів
	1	2

КПІ ім. Ігоря Сікорського, ФПМ, КВ-51

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.006 Д2	Система управління	1		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Схема алгоритму			
	А1	ІАЛЦ.045490.008 Д4	Система управління	1		
			програмованими			
			пристроями взаємодії з			
			користувачем			
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.			
			Графічний матеріал.			
			Презентація.			

					ІАЛЦ.045490.003 ТП	Арк.
						2
Змін.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ДОДАТКИ

1. ІАЛЦ.045490.005 Д1. Взаємодія зовнішніх пристроїв та ПК за допомогою USB інтерфейсу. Схема структурна
2. ІАЛЦ.045490.006 Д2. Інтерфейс USB HID бібліотеки. Схема структурна
3. ІАЛЦ.045490.007 Д3. Монітор HID пристроїв. Блок-схема алгоритму
4. ІАЛЦ.045490.008 Д4. Структура кросплатформенної HID бібліотеки. Діаграми класів.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

КМ – комп’ютерна мережа

КС – комп'ютерна система

ОС – операційна система

ПЗ – програмне забезпечення

ПСІ – прикладний програмний інтерфейс

ПКП – програмований користувацький пристрій

СВВ – система вводу-виводу

HID (Human Interface Device) – пристрої взаємодії з користувачем

ISDN (Integrated Services Digital Network) - цифрова мережа з інтегрованими службами

USB (Universal Serial Bus) – універсальна послідовна шина

WDK (Windows Driver Kit) - набір програмних засобів від Microsoft, що дозволяє розробляти ПЗ для пристроїв для платформи Microsoft Windows.

ВСТУП

В наш час більшість периферійних пристроїв підтримують інтерфейс USB, особливо часто вони використовуються з пристроями, які побудовані на мікропроцесорах. Цей інтерфейс дозволяє не тільки обмінюватися даними, а й забезпечує електроживлення периферійного пристрою, чим не забули скористатися розробники.

На сьогоднішній день ринок USB пристроїв є дуже різноманітним і включає в себе такі електричні пристрої, як USB-капці з підігрівом, підігрівач для чашки, вентилятор, ліхтарик, грілку та навіть зубну щітку. Існує навіть швейцарський армійський ніж, в який вбудовано флеш-накопичувач з портом USB. Іншими більш звичними USB пристроями, які використовуються для взаємодії користувача з персональним комп'ютером є клавіатура, мишка та різноманітні контролери.

Сучасні користувацькі пристрої відрізняються від своїх попередників, які використовували повільніші та менш гнучкі інтерфейси та мали мінімальні базові функції для їх застосування. Останнім часом особливої популярності набирає ігрова периферія, що використовує різнокольорову підсвітку, за допомогою вбудованих RGB діодів, та додаткові клавіші, для забезпечення додаткових можливостей або такі, функціонал яких можна визначати самостійно.

Метою даного дипломного проекту є розробка системи взаємодії з користувацькими пристроями, що базуються на класі HID для обміну даними між ПК та користувацькими пристроями. Система повинна бути виконана у вигляді додатку з графічним інтерфейсом, що підтримується найпопулярнішими сучасними операційними системами – Windows, Linux, MacOS та відображати загальну інформацію про наявні у системи пристрої, що працюють через HID інтерфейс.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Особливості взаємодії сучасної переферії з сучасними операційними системами

USB (універсальна послідовна шина) - це інтерфейс і протокол, який дозволяє одному головному комп'ютеру (хосту) спілкуватися з одним або множиною периферійних пристроїв. Специфікація протоколу USB визначає цей інтерфейс.

Згідно з специфікацією USB, пристрої можуть бути хабами, функціями або їх комбінацією.

Пристрій-хаб (hub) лише забезпечує додаткові точки підключення пристроїв до шини.

Пристрій-функція (function) USB надає системі додаткові функціональні можливості, наприклад, підключення до ISDN, цифровий джойстик, акустичні колонки з цифровим інтерфейсом і т. п.

Комбінований пристрій (compound device), який містить декілька функцій, подається як хаб з підключеними до нього декількома пристроями.

Кожен USB пристрій повинен мати інтерфейс USB, для забезпечення повної підтримки його протоколу, виконання стандартних операцій (конфігурація і скидання) і подання інформації, яка описує пристрій. Роботою всієї системи USB керує хост-контролер, який є програмно-апаратною підсистемою хоста. Згідно зі специфікацією більшість USB пристроїв потребують програмного інтерфесу зі сторони хоста та програмного забезпечення самого пристрою (firmware). Комунікація через протокол USB відбувається безпосередньо між хостом та пристроєм, де хост здійснює контроль над шиною та весь час ініціює з’єднання, за винятком тих пристроїв, які мають функцію самостійного віддаленого пробудження [1][2].

У порівнянні з іншими інтерфейсами USB пропонує безліч переваг, до яких входить мінімальна кількість ліній переривання (IRQ Lines), автоматична конфігурація перерахунку підключень, низька вартість, низьке енергоспоживання, висока швидкість та надійність. З'єднання через цей протокол дозволяє підключати та відключати пристрої під час роботи хост-комп'ютера (без знеструмлення шини).

Шина USB є хост-центровою: єдиним ведучим пристроєм, який керує обміном є хост-комп'ютер, а всі приєднані до неї периферійні пристрої – виключно залежними.

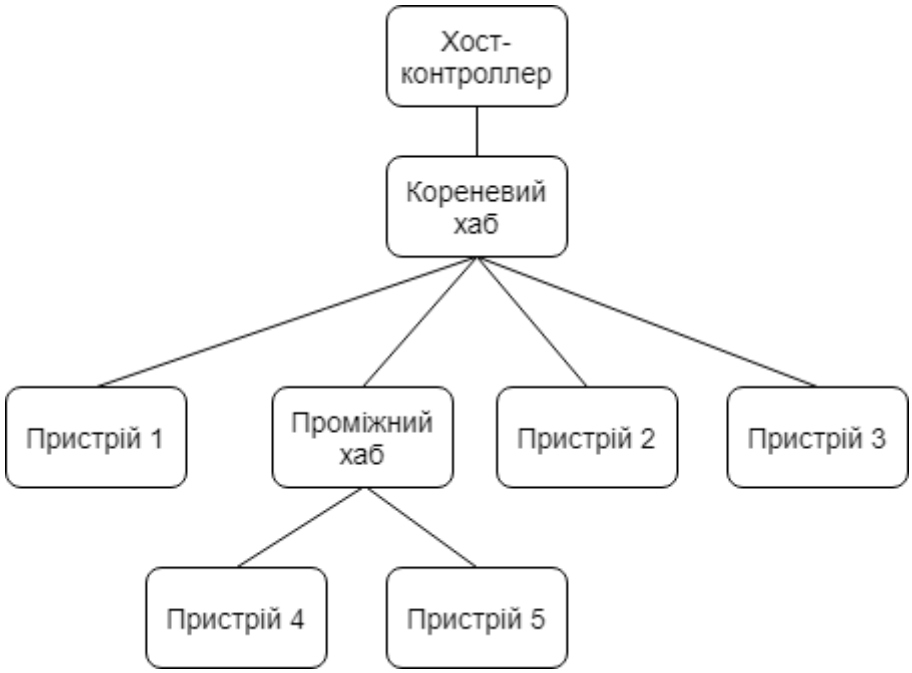


Рисунок 1.1 – Фізична топологія USB шини

Фізична топологія шини USB – багатоярусна зірка. Її вершиною є хост-контролер, об'єднаний з кореневим хабом (root hub), який, як правило, має два або більше портів для підключення. Хаб є пристроєм-розгалужувачем, він може бути і джерелом живлення для підключених до нього пристроїв. До кожного порту хаба може безпосередньо підключатись периферійний пристрій або проміжний хаб; шина допускає до п'яти рівнів каскадів хабів (не враховуючи кореневого) [3]. Оскільки комбіновані

пристрої всередині себе місять хаб, їх підключення до хабу шостого ярусу вже недопустиме. Проміжні хаби мають декілька низхідних (downstream) портів для підключення периферійних пристроїв і один висхідний (upsteam) порт для підключення до кореневого хаба або низхідного порту вищого за ієрархією хаба. Приклад фізичної топології зображено на рис. 1.1.

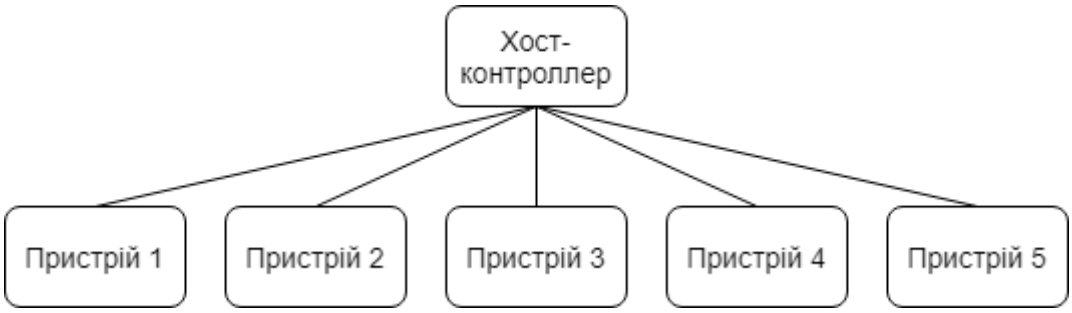


Рисунок 1.2 – Логічна топологія USB шини

Логічна топологія USB – просто зірка: для хост-контролера хаби створюють ілюзію безпосереднього підключення кожного пристрою. На відміну від шин розширення (ISA, PCI, PC Card), де програма взаємодіє з пристроями за допомогою звернень за фізичними адресами комірок пам’яті, портів введення-виведення, переривань і каналів DMA, взаємодія додатків з пристроями USB виконується лише через програмний інтерфейс. Цей інтерфейс, який забезпечує незалежність звернень до пристроїв, надається системним ПЗ контролера USB. Приклад фізичної топології зображено на рис. 1.2 [5][6].

Кожен пристрій на шині USB (їх може бути до 127) при підключенні автоматично отримує свою унікальну адресу. Логічно пристрій являє собою набір незалежних кінцевих точок (endpoint, EP), з якими хост-контролер (і клієнтське ПЗ) обмінюються інформацією. Кожна кінцева точка має свій номер і описується такими параметрами:

- необхідна частота доступу до шини і допустимі затримки

обслуговування;

- необхідна смуга пропускання каналу;
- вимоги до обробки посилок;
- максимальні розміри переданих і прийнятих пакетів;
- тип передавання;
- напрям передавання

Кожний пристрій обов’язково містить кінцеву точку з номером 0, яка використовується для ініціалізації, загального управління і опитування стану пристрою. Ця точка завжди встановлена при включенні живлення і підключенні пристрою до шини. Вона підтримує передавання типу «керування».

Окрім нульової точки, пристрої-функції можуть мати додаткові точки, які реалізовують корисний обмін даними. Низькошвидкісні пристрої можуть мати до двох додаткових точок, повношвидкісні – до 15 точок введення і 15 точок виведення (протокольне обмеження). Додаткові точки (лише вони надають корисні для користувача функції) не можуть бути використані до їх конфігурування (встановлення узгодженого з ними каналу) [7].

Каналом (pipe) в USB називається модель передавання даних між хост-контролером і кінцевою точкою пристрою. Існують два типи каналів: потоки і повідомлення.

Потік (stream) доставляє дані від одного кінця каналу до іншого, він завжди однонаправлений. Один і той же номер кінцевої точки може використовуватись для двох потокових каналів – введення і виведення. Потік може реалізовувати такі типи обміну: передавання масивів, ізохронний і переривання.

Повідомлення (message) має формат, визначений специфікацією USB. Хост надсилає запит до кінцевої точки, після якого передається

(приймається) пакет повідомлення, за яким слідує пакет з інформацією стану кінцевої точки.

Наступне повідомлення нормально не може бути надіслане до обробки попереднього, але при обробці помилок можливий скид не обслужених повідомлень. Двосторонній обмін повідомленнями адресується до однієї і тієї ж кінцевої точки.

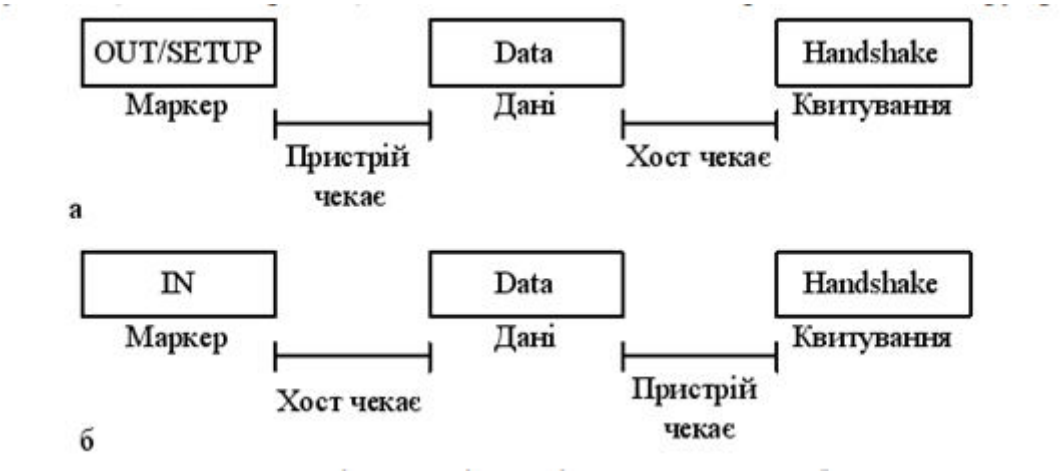


Рисунок 1.3 – Послідовність пакетів в транзакціях

Всі обміни (транзакції) з пристроями USB складаються з двох-трьох пакетів. Кожна транзакція планується і починається за ініціативою контролера, який надсилає пакет-маркер (token packet). Він описує тип і напрямок передавання, адресу пристрою USB і номер кінцевої точки. В кожній транзакції можливий обмін лише між кінцевою точкою і хостом. Адресований маркером пристрій розпізнає свою адресу і готується до обміну. Джерело даних (визначене маркером) передає пакет даних (або повідомлення про відсутність даних, призначених для передавання). Після успішного приймання пакета приймач даних відсилає пакет квитування (handshake packet)[6]. Послідовність пакетів в транзакціях ілюструє рис. 1.3.

Для виявлення помилок передавання кожний пакет має контрольні поля CRC-кодів, які дозволяють виявляти всі одинарні і подвійні бітові

помилки. Апаратні засоби виявляють помилки передавання, а контролер автоматично виконує трикратну спробу передавання. Якщо повторювання безуспішні, повідомлення про помилку передається клієнтському ПЗ.

Всі подробиці організації транзакцій від клієнтського ПЗ ізолюються контролером USB і його системним програмним забезпеченням.

В залежності від програми, може бути обраний один (або більше) з чотирьох типів передачі даних:

- Керуючі посилки (control transfers) використовуються для конфігурування пристроїв під час їх підключення і для управління пристроями в процесі роботи. Протокол забезпечує гарантовану доставку даних.
- Передавання масивів даних (bulk data transfers) – це передавання без будь-яких зобов'язань щодо затримки доставки і швидкості передавання. Передавання масивів можуть займати всю смугу пропускання шини, вільну від передавання інших типів. Пріоритет цього передавання найнижчий, вони можуть призупинятись при сильному завантажені шини. Доставка гарантована – при випадковій помилці виконується повтор. Передавання масивів доречні для обміну даними з принтерами, сканерами, пристроями зберігання і т. п.
- Переривання (interrupt) – короткі передачі, які мають спонтанний характер і повинні обслуговуватись не повільніше, ніж того потребує пристрій. Межа часу обслуговування встановлюється в діапазоні 10 – 225 мс для низької, 1 – 225 мс для повної швидкості, для високої швидкості можна замовити і 125 мкс. При випадкових помилках обміну виконується повтор. Переривання

використовуються, наприклад, при введені символів з клавіатури або для передавання повідомлення про переміщення миші.

- Ізохронні передачі (isochronous transfers) – неперервні передачі в реальному часі, які займають попередньо узгоджену частину пропускної здатності шини з гарантованим часом затримки доставки. Дозволяють на повній швидкості організувати канал зі смугою 1,023 Мбайт/с (або два по 0,5 Мбайт/с), зайнявши 70% доступної смуги (залишок можна заповнити і менш ємнісними каналами). На високій швидкості кінцева точка може отримати канал до 24 Мбайт/с (192 Мбіт/с). У випадку виявлення помилки ізохронні дані не повторюються – недійсні пакети ігноруються. Ізохронні передачі потрібні для потокових пристроїв: відеокамер, цифрових аудіопристроїв (колонки USB, мікрофон), пристроїв відтворення і запису аудіо- і відеоданих (CD і DVD). Відеопотік (без компресії) шина USB має можливість передавати лише на високій швидкості.

1.2 Загальні проблеми розробки систем управління користувацькими пристроями

Універсальна послідовна шина є сучасним стандартом для з'єднання переферії з комп'ютером, щороку продаються сотні мільйонів пристроїв з підтримкою даного протоколу. Такий шалений успіх звичайно завдячує й кінцевому продукту, що використовує USB. Розробники даних продуктів в багатьох випадках, вимушені створювати власні USB драйвера для ппідтримки їх пристроїв. На щастя, є більш простіший спосіб додати підтримку USB-пристроїв.

Для того, щоб уникнути проблем створення власних USB драйверів для взаємодії з користувацькими пристроями, можна використати стандартний HID клас USB пристроїв, який за замовчуванням підтримується всіма сучасними операційними системами.

HID (Human Interface Device) - це клас пристроїв для взаємодії з користувачем. До цього класу, як правило, відносять наступні пристрої:

- Клавіатури
- Мишки
- Різноманітні контролери (включаючи аналогові та цифрові)
- LED виходи (світлодіоди)

Основні особливості і обмеження HID пристроїв:

- Повношвидкісні HID пристрої передавати до 64 000 байт за секунду (64 байт/мс). Для низькошвидкісних пристроїв встановлена швидкість передачі даних тільки 800 байт за секунду (0.8 байт/мс).
- HID пристрій може задати частоту свого опитування для встановлення, чи є нові дані для пересилання
- Весь обмін з HID пристроєм відбувається за допомогою певної структури, яка називається звітом (Report). Один звіт може вміщувати до 65 535 байт даних. Він має достатньо гнучку структуру для описання любого типу пристрою і формату передачі даних.
- Оскільки всі сучасні ОС мають вбудовану підтримку HID класу пристроїв, то зникає необхідність в трудомісткій розробці власного драйверу для роботи з користувацькими пристроями.

1.3 Аналіз існуючих засобів для управління програмованими пристроями

Існують спеціалізовані бібліотеки для доступа до шини USB.

Розглянемо деякі з них.

Бібліотека HID USB Library - бібліотека написана на мові C#, поширюється у вигляді динамічної бібліотеки (DLL). Бібліотека надає можливість працювати тільки з HID-пристроями.

Бібліотека WinUSB - це пропрієтарний USB-драйвер, який поставляється корпорацією Microsoft. Він дозволяє отримати прямий доступ до пристрою тільки від одного додатка. Поширюється у вигляді DLL-бібліотеки. WinUSB дозволяє вести обмін даними з USB-пристроєм через кінцеві точки. До основних переваг можна віднести відсутність необхідності розробляти власний драйвер і, як наслідок, прискорення швидкості розробки. До недоліків відносять те, що тільки один додаток має доступ до пристрою в будь-який момент, підтримка ізохронної передачі з'явилася, починаючи з Windows 8.1 та обмеження інтерфейсу в конфігурації USB- пристрою.

Бібліотека HidApi - бібліотека, яка надає можливість для створення програмного забезпечення з доступом до інтерфейсів пристроїв USB і Bluetooth на операційних системах Linux, Windows, Mac OS. HidApi може використовуватися як для обміну даними зі стандартними пристроями HID, так і для роботи з користувацькими пристроями. HidApi надає тестовий додаток, який може повертати список пристроїв, що підключеним до операційної системи,. і обмінюватися даними з будь-яким них.

Бібліотека LibUsb - кросплатформенна бібліотека, яка надає доступ до USB-пристроїв, вона містить всі необхідні драйверами і бібліотеки функцій для роботи в операційних системах Windows і Linux. Ця бібліотека дуже популярна і дозволяє швидко розробляти програми, які взаємодіють з пристроями за допомогою стандартних функцій. Це виключає необхідність написання власного драйвера пристрою, що істотно

заощаджує час.

2. СТРУКТУРА ПРОГРАМНИХ ЗАСОБІВ

2.1 Опис інструментарію

Для створення додатку з графічним інтерфейсом, який відповідає відповідати всім вище перерахованим вимогам, використано бібліотеки Qt. Qt Framework – крос-платформенний інструментарій для розробки програмного забезпечення мовою програмування C++, який також надає можливість для його використання з багатьма іншими мовами програмування (Python, Ruby, Java, Go та багато інших).

Бібліотека Qt дозволяє запускати написане з її допомогою програмне забезпечення на більшості сучасних операційних системах шляхом звичайної перекомпіляції програми під певну архітектуру без зміни коду програми. Вона включає в себе всі основні класи, які можуть знадобитися для розробки програмного забезпечення, починаючи з елементів графічного інтерфейсу і закінчуючи класами для роботи інтернет мережею, базами даних, OpenGL, SVG та XML. Qt є повністю об’єктно-орієнтованим, може бути легко доповненим та підтримує техніку компонентного програмування [23].

Таким чином, використання цієї бібліотеки дозволяє створити застосунок, що працюватиме та матиме однакову поведінку під час роботи з багатьма сучасними операційниими системами, такими як Windows, Linux та MacOS.

З моменту свого створення в 1996 році комерційна версія бібліотеки Qt лягла за основу тисячі успішних проектів у всьому світі. Крім того, Qt є фундаментом популярного робочого середовища KDE, що входить в склад дистрибутивів GNU/Linux.

Першим значним кроком у напрямі збільшення популярності та відкритості Qt, стала зміна на початку 2009 року ліцензії з GPL на LGPL 2.1, що дозволило безперешкодно використовувати Qt в закритих проектах

без необхідності купівлі комерційної ліцензії або відкриття сирцевих текстів свого продукту під ліцензією GPL.

До основних переваг Qt, які стали ключовими під час вибору даної бібліотеки для створення користувацького додатку, можна віднести:

- Об’єктно-орієнтоване програмування. Парадигма програмування, в якій базовими концепціями є поняття об’єктів та класів. Клас – це тип, що описує характеристики екземплярів об’єктів, що інкапсулює (включає в себе) як дані, так і процедури для їх обробки. Можливе як наслідування даних та процедур, так і їх поліморфізм (зміна). Об'єктно-орієнтоване програмування в даний час є абсолютним лідером в області прикладного програмування. У той же час в області системного програмування досі лідирує суто процедурний мову С, хоча при взаємодії системного і прикладного рівнів операційних систем помітний вплив стали надавати мови об'єктно-орієнтованого програмування. Тому Qt, написана на мові С ++, стала своєрідним крос-платформенним стандартом.
- Велика кількість вбудованих функцій, необхідних для чисельного моделювання. Так, наприклад, в бібліотеці вже визначені класи векторів, матриць і операції над ними.
- Потужні і зручні засоби для розробки графічного інтерфейсу користувача, що включають в себе всі необхідні стандартні елементи, які до того ж можуть бути легко модифіковані для конкретної програми.
- Інтеграція в різні середовища розробки програмного забезпечення такі як Microsoft Visual Studio та Dev-Cpp.
- Простота і гнучкість програмування, заснована на концепції слотів і сигналів, що підвищує ефективність і швидкість розробки програми.

Сигнали і слоти використовуються для зв'язку між об'єктами Qt. Механізм сигналів і слотів є найважливішою здатністю Qt і, можливо, тим,

чим Qt найбільше відрізняється від інших інструментаріїв, які часто використовують механізми зворотного зв'язку (callback). Сигнали виходять від об'єктів, коли відбувається якась подія (наприклад, користувач друкує щось в поле введення або натискає кнопку). Слот - це функція, що викликається у відповідь на певний сигнал. Засоби Qt дозволяють просто зв'язати сигнал і відповідний йому слот [23].

2.2 Загальна структура HID протоколу

Специфікація HID класу визначає основні вимоги і протокол передачі даних і для цього користувачу не обов’язково безпосередньо взаємодіяти з пристроєм. Пристрої класу HID повинні відповідати декільком вимогам, які накладаються на них, щоб HID-інтерфейс був стандартизований та ефективний:

- Всі HID пристрої мусять мати кінцеву точку контролю (Endpoint 0) і кінцеву точку переривання (IN). Багато пристроїв також використовують точку переривання (OUT). У більшості випадків HID-пристроям не дозволяється мати більше однієї кінцевої точки OUT і однієї IN.
- Всі передані дані повинні бути відформатовані як звіти (Reports), структура яких визначена в дескрипторі звіту. Опис звіту буде наведений далі цьому розділі.
- Пристрої HID повинні відповідати на стандартизовані запити HID на додачу до всіх стандартизованих запитів USB.

Перед тим, як HID-пристрій зможе перейти всій робочий режим і та приступити до обміну даними з хостом, пристрій повинен проініціалізуватися да додатися до списку визначених в системі пристроїв. Процес визначення (енумерації) складається з ряду викликів, зроблених хостом за допомогою дескриптора, що описує можливості пристрою.

Пристрій повинен повністю відповідати дескриптору, який

відповідає стандартизованому формату. Дескриптори містять всю основну інформацію про пристрій. Специфікація USB визначає деякі з отриманих дескрипторів, а специфікація HID визначає інші необхідні дескриптори.

Дескриптор повинен починатися з байту, що описує його довжину байтах. Ця довжина дорівнює загальній кількості байтів в дескрипторі, включаючи байт, що зберігає довжину. Наступний байт вказує тип дескриптора, який дозволяє хоста правильно інтерпретувати інші байти, що містяться в ньому. Вміст і значення інших байтів специфічні для типу дескриптора, який передається. Структура дескриптора повинна точно відповідати специфікаціям, хост буде ігнорувати отримані дескриптори, що містять помилки в розмірі або значенні, потенційно викликаючи переривання і не дозволяючи подальше спілкування між пристроєм і хостом.

```
typedef struct
{
   BYTE bLength;                // Size of this Descriptor in Bytes
   BYTE bDescriptorType;        // Descriptor Type (=1)
   WORD bcdUSB;                 // USB Spec Release Number in BCD
   BYTE bDeviceClass;           // Device Class Code
   BYTE bDeviceSubClass;        // Device Subclass Code
   BYTE bDeviceProtocol;        // Device Protocol Code
   BYTE bMaxPacketSize0;        // Maximum Packet Size for EP0
   WORD idVendor;               // Vendor ID
   WORD idProduct;              // Product ID
   WORD bcdDevice;              // Device Release Number in BCD
   BYTE iManufacturer;          // Index of String Desc for Manufacturer
   BYTE iProduct;               // Index of String Desc for Product
   BYTE iSerialNumber;          // Index of String Desc for SerNo
   BYTE bNumConfigurations;     // Number of possible Configurations
} device_descriptor;            // End of Device Descriptor Type
```

Рисунок 2.1 – Визначення типу дескриптора пристрою

Приклад структури дескриптора вказано на рис 2.1. Це визначення відповідає вимогам специфікації USB для розмір та порядку вмісту дескриптора пристрою. Деякі полям містять всього один байт (BYTE) для збереження необхідної інформації, інші – використовують для цього два

байти (WORD).

```
const code device_descriptor DeviceDesc =
{
 18,                 // bLength
 0x01,               // bDescriptorType
 0x1001,             // bcdUSB
 0x00,               // bDeviceClass
 0x00,               // bDeviceSubClass
 0x00,               // bDeviceProtocol
 EP0_PACKET_SIZE,    // bMaxPacketSize0
 0xC410,             // idVendor
 0x0001,             // idProduct
 0x0000,             // bcdDevice
 0x01,               // iManufacturer
 0x02,               // iProduct
 0x00,               // iSerialNumber
 0x01                // bNumConfigurations
}; //end of DeviceDesc
```

Рисунок 2.2 – Приклад можливого наповнення дескриптора

Приклад можливого наповнення дескриптора вказаний на рис. 2.2, він відповідає всім вимогам структури device_descripror наведеної на рис. 2.1. Весь визначені в ньому вміст повинен бути правильним і перевіреним ще під час компіляції системи вбудованого програмного забезпечення, оскільки всі ці дані зберігаються в енергонезалежній постійній пам’яті. Вмісту полів дескриптора, розмір типу яких є більше одного байту повинен зберігатися в стилі Little Endia, при якому першим записується найменш значущий байт. Наприклад, десяткове значення 300 з шістнадцятковим представленням 0x012C, буде збережено в пам’яті як 0x2C01.

Значення поля bcdUSB визначає версію специфікації USB, формату

якої пристрій і його дескриптори відповідають. Якщо значення версії вказане десятковим числом, то верхній байт являє собою ціле число, наступні чотири біти - десяті, а останні чотири біти - соті. USB 1.1 - 0110h (не 0101h). USB 2.0 - 0200h. USB 3.0 - 0300h. Пристрій зі значенням поля bcdUSB = 0210h або вище повинен підтримувати дескриптор BOS. Пристрій або адаптер дроту, який відповідає стандарту Wireless USB V1.0, повинен мати поле bcdUSB зі значенням до 0250h [16][17].

Значення	Опис класу
00h	Дескриптор інтерфейсу визначає клас, а функція не використовує дескриптор асоціації інтерфейсів. (Див. EFh нижче.)
02h	Пристрій зв'язку (може замість цього бути оголошено на рівні інтерфейсу)
09h	Хаб
0Fh	Персональний пристрій охорони здоров'я (бажано оголошувати на рівні інтерфейсу)
DCh	Пристрій для діагностики (може замість того, щоб бути оголошений на рівні інтерфейсу) bDeviceSubclass = 01h, bDeviceProtocol = 01h: Пристрій, що відповідає USB 2.0
E0h	Бездротовий контролер (може бути оголошений на рівні інтерфейсу) bDeviceSubclass = 01h: інтерфейс програмування Bluetooth
EFh	Різне bDeviceSubclass = 01h bDeviceProtocol = 01h: активна синхронізація bDeviceProtocol = 02h: пасивна синхронізація

	bDeviceSubclass = 02h bDeviceProtocol = 01h: дескриптор асоціації інтерфейсів bDeviceProtocol = 01h: дротовий адаптер багатофункціональний периферійний (Wireless USB).
FFh	Залежний від постачальника (замість того, щоб бути оголошена на рівні інтерфейсу)

Таблиця 2.1 – Перелік класів, які використовуються в полі bDeviceClass дескриптора пристроїв

Поле bDeviceClass дескриптора визначає клас пристрою для яких, функції визначається на рівні пристрою. Значення від 01h до FEh є зарезервованими для класів, визначених специфікацією USB (табл. 1.1). Класи, функції яких визначені виробником, використовують значення FFh. Більшість пристроїв вказують їх клас або класи в дескрипторах інтерфейсу. Для цих пристроїв bDeviceClass в дескрипторі пристрою дорівнює 00h, якщо функція не використовує дескриптор асоціації інтерфейсу або EFh, якщо функція використовує дескриптор асоціації інтерфейсів.

Поле bDeviceSubclass може вказати підклас всередині класу. Підклас може додавати підтримку для додаткових функцій і можливостей, що визначаються групою функцій класу. Якщо значення bDeviceClass дорівнює 00h, bDeviceSubclass також має дорівнювати 00h. Якщо bDeviceClass знаходиться в діапазоні 01h – FEh, bDeviceSubclass дорівнює 00h або коду, визначеному для класу пристрою. Підкласи, визначені виробником у стандартних класах, використовують FFh.

Поле bDeviceProtocol задає протокол для вибраного класу і підкласу. Наприклад, USB 2.0 хаб використовує це поле, щоб вказати, чи він наразі підтримує високу швидкість передачі даних, і якщо так, то чи хаб

підтримує один або декілька перекладачів транзакцій. Якщо bDeviceClass знаходиться в діапазоні 01 – FEh, то значення протоколу має дорівнювати 00h або коду, визначеному класом пристрою.

Значення поля bMaxPacketSize0 вказує на максимальний розмір пакета для нульової точки. Хост використовує цю інформацію в запитах, які слідують запиту дескриптора пристрою. Для USB 2.0 максимальний розмір пакета дорівнює значенню поля і повинен бути 8 байт для низьких швидкостей; 8, 16, 32 або 64 для повної швидкості; і 64 для високої швидкості. Для супер швидкісної максимальний розмір пакета має дорівнювати значенню поля 2bMaxPacketSize0, а bMaxPacketSize0 повинен дорівнювати 9, щоб визначити максимальний розмір пакета 512.

Поле idVendor визначає ідентифікатор постачальника, що надається представниками USB-IF (з англ. USB Implementers Forum – форум розробників USB) членам їхнього форуму та іншими, хто сплачує адміністративний збір. Хост може мати INF-файл, який містить це значення, і якщо це так, Windows може використовувати значення, щоб допомогти вибрати драйвер для пристрою. За винятком пристроїв, які використовуються тільки в будинку, де користувач відповідає за запобігання конфліктів, кожен дескриптор пристрою повинен мати дійсний ідентифікатор постачальника в цьому полі.

Поле idProduct - це ідентифікатор продукту, який визначає пристрій постачальника. Власник ідентифікатора постачальника визначає ідентифікатор продукту. Дескриптор пристрою і INF-файл пристрою на хості можуть містити це значення, і якщо це так, Windows може використовувати його, щоб допомогти вибрати знайти та використати драйвер для пристрою. Кожен ідентифікатор продукту є унікальним тільки для одного постачальника, тому кілька постачальників можуть використовувати той самий ідентифікатор продукту без конфлікту.

Значення поля bcdDevice визнає номер випуску пристрою у форматі BCD. Виробник сам визначає це значення. Хост, як правило, використовує це значення, щоб допомогти вибрати драйвер для пристрою.

Значення поля iManufacturer – це індекс відмінний від нуля, який вказує на рядок, що описує виробника або нулевий, якщо дескриптор виробника відсутній.

Значення поля iProduct - це індекс відмінний від нуля, який вказує на рядок, що описує продукт, або нулевий, якщо дескриптор цього рядка.

Значення поля iSerialNumber визначає індекс, який вказує на рядок, що містить серійний номер пристрою або 00h, якщо немає серійного номера. Серійні номери корисні, якщо користувачі можуть мати більше одного ідентичного пристрою на шині, і хост повинен розрізняти кожен з цих пристроїв навіть після перезавантаження. Серійний номер також дозволяє хосту визначати, чи є пристрій тим самим, який раніше використовувався, чи це новий пристрій з ідентичним ідентифікатором постачальника та ідентифікатором продукту. Пристрої з ідентичним ідентифікатором постачальника, ідентифікатором продукту та номером випуску пристрою не повинні використовувати спільний серійний номер. Запам’ятовуючі пристрої, що використовують протокол об'ємного зберігання, повинні мати серійні номери.

Значення поля bNumConfigurations дорівнює кількості конфігурацій, які підтримує пристрій при поточній робочій швидкості.

Дескриптор звіту не схожий на інші дескриптори, оскільки це не просто таблиця значень. Тривалість і вміст дескриптора звіту змінюються залежно від кількості полів даних, необхідних для звіту чи звітів пристрою. Дескриптор звіту складається з елементів, які надають інформацію про пристрій. Перша частина елемента містить три поля: тип

елемента, тег елемента та розмір елемента. Разом ці поля визначають тип інформації, яку надає елемент [19].

Існує три типи елементів: основний, глобальний та локальний. Наразі визначено п'ять основних тегів:

- Мітка вхідного елемента: стосується даних з одного або декількох подібних елементів керування на пристрої. Наприклад, змінні дані, такі як зчитування положення однієї осі або групи важелів або масивів даних, таких як одна або більше натискання кнопок або перемикачів.
- Вихідний тег елемента: Відноситься до даних до одного або декількох аналогічних елементів управління на пристрої, таких як встановлення положення однієї осі або групи важелів (змінних даних). Або він може представляти дані до одного або більше світлодіодів (даних масиву).
- Тег елемента функції: описує вхід і вихід пристрою, не призначений для споживання кінцевим користувачем, наприклад, функцію програмного забезпечення або панель керування.
- Тег елемента збору: значуще групування елементів введення, виводу та функції - наприклад, миші, клавіатури, джойстика і покажчика.
- Тег кінцевого елемента збору: кінцевий елемент, що використовується для визначення кінця набору елементів.

Дескриптор звіту надає опис даних, наданих кожним елементом керування в пристрої. Кожен тег головного елемента (Input, Output або Feature) ідентифікує розмір даних, що повертаються певним контролем, і визначає, чи є дані абсолютними або відносними, та іншу відповідну інформацію. Попередні локальні та глобальні елементи визначають

мінімальні та максимальні значення даних тощо. Дескриптор звіту - це повний набір всіх елементів для пристрою. Дивлячись на дескриптор звіту, програма знає, як обробляти вхідні дані, а також те, для чого можна використовувати дані.

Одне або більше полів даних з елементів управління визначаються головним елементом і далі описуються попередніми глобальними і локальними елементами. Місцеві елементи описують лише поля даних, визначені наступним головним елементом. Глобальні елементи стають атрибутами за замовчуванням для всіх наступних полів даних цього дескриптора.

Дескриптор звіту може мати кілька основних елементів. Він повинен містити кожен із наведених нижче елементів для опису даних керування (всі інші елементи є необов'язковими):

- Введення (вихід або функція)
- Використання
- Сторінка використання
- Логічний мінімум
- Логічний максимум
- Розмір звіту
- Повідомлення про кількість

```
Usage Page (Generic Desktop),           ;Use the Generic Desktop Usage Page
Usage (Mouse),
    Collection (Application),           ;Start Mouse collection
    Usage (Pointer),
    Collection (Physical),              ;Start Pointer collection
      Usage Page (Buttons)
      Usage Minimum (1),
      Usage Maximum (3),
      Logical Minimum (0),
      Logical Maximum (1),              ;Fields return values from 0 to 1
      Report Count (3),
      Report Size (1),                  ;Create three 1 bit fields (button 1, 2, & 3)
      Input (Data, Variable, Absolute), ;Add fields to the input report.
      Report Count (1),
      Report Size (5),                  ;Create 5 bit constant field
      Input (Constant),                 ;Add field to the input report
      Usage Page (Generic Desktop),
      Usage (X),
      Usage (Y),
      Logical Minimum (-127),
      Logical Maximum (127),            ;Fields return values from -127 to 127
      Report Size (8),
      Report Count (2),                 ;Create two 8 bit fields (X & Y position)
      Input (Data, Variable, Relative), ;Add fields to the input report
    End Collection,                     ;Close Pointer collection
End Collection                          ;Close Mouse collection
```

Рисунок 2.3 – Приклад звіту для мишки з трьома клавішами

На рис. 2.3 наведено зразок кодування значень полів для визначення миші з 3 кнопками. У цьому випадку головним елементам передують глобальні елементи, такі як використання, кількість звітів або розмір звіту (кожен рядок є новим).

2.3 Особливості взаємодії з HID пристроями в середовищах ОС Windows, Linux, MacOS

Кожна сучасна операційна система має вбудовану підтримку пристроїв, які працюють через HID протокол. Для того, щоб цього досягти було розроблено спеціальні засоби для взаємодії з користувацькими USB пристроями, які входять в стандартний пакет під час їх встановлення ОС.

Операційна система Windows надає відкрите API, яке містить функції та програми, що можуть бути використані для отримання інформації про звіти HID-пристроїв, а також зчитувати та записувати дані за допомогою

цих звітів. В Windows DDK надає всю необхідну специфікацію для використання цих функції.

Windows API надає ряд функцій, які дозволяють додаткам знайти всі пристрої в класі пристрою та отримати шлях до пристрою для кожного пристрою. Функція CreateFileA може використовувати шлях до пристрою, щоб отримати дескриптор для доступу до пристрою.

Windows HID API, за замовчуванням передбачає, що кожен елемент звіту є кнопкою або значенням. Як визначено в HID API, кнопка є елементом керування або даними, що мають дискретне, двійкове значення, наприклад, ON (1) або OFF (0). Використання значення або значення може мати будь-який діапазон значень. Будь-який елемент звіту, який не є кнопкою, є числовим значенням.

Отримання шляху до пристрою вимагає наступних кроків:

1. Отримати GUID інтерфейсу пристрою.
2. Запит вказівника на набір інформації про пристрій з інформацією про всі встановлені та наявні пристрої в класі інтерфейсу пристрою.
3. Запит вказівника на структуру, яка містить інформацію про інтерфейс пристрою в наборі інформації про пристрій.
4. Запит структуру, яка містить ім'я шляху пристрою інтерфейсу пристрою.
5. Отримання назви пристрою зі структури.

Після цього програма може використовувати шляху пристрою, щоб відкрити обробник для встановлення зв’зку з пристроєм.

В операційній системи Linux за контроль підключення та видалення пристроїв відповідає ядро системи. Зміни стану пристроїв (підключення нового або видалення існуючого) повинні бути при цьому видимі в користувацькому просторі. При підключенні нових пристроїв вони повинні бути перевірені належним чином та (при необхідності)

визначитися користувальницькими додатками. Якщо користувацьке ПЗ працює з конкретним пристроєм, то його необхідно інформувати про будь-яке зміни стану пристрою. Стандартна бібліотека udev забезпечує всі необхідні засоби для динамічного створення і видалення файлів пристроїв і символічних посилань в каталозі /dev.

Правила які можуть задаватися під час використання ПЗ udev дозволяють використовувати зовнішні програми для обробки подій ядра обчислень пристроїв (ядра пристроїв подій), що дозволяє змінювати порядок дій, наприклад, написання власних скриптів або запитів і додаткових даних для використання в процесі роботи.

Також разом з бібліотекою udev в стандартний пакет Linux додається бібліотека libudev, яка надає розробникам програмного забезпечення можливість користуватися відкритим API для взаємодії з USB пристроями, наявними в операційній системі.

Всі функції вимагають використання контексту libudev для їх роботи. Цей контекст можна створити за допомогою функції udev_new. Він використовується для відстеження стану бібліотеки та об'єднання об'єктів.

Бібліотека надає можливість виділяти кілька незалежних об'єктів і використовувати кожен своєму потоці паралельно. Однак, не можна виділяти такий об'єкт в одному потоці, а також працювати або звільняти його вже в іншому, навіть при використанні блокувань, щоб ці потоки не працювали на ньому одночасно.

Для того, щоб знайти локальний пристрій в системі, за допомогою об'єкту пристрою udev можна створити за допомогою udev_device_new_from_syspath. Об'єкт пристрою дозволяє запитувати атрибути поточного стану, атрибути зчитування, запису та властивості пошуку відповідного пристрою.

Для перерахування локальних пристроїв у системі може бути

створений об'єкт перерахування за допомогою функції udev_enumerate_new.

Для моніторингу подій підключення або відключених пристроїв може бути створений об'єкт монітору за допомогою функції udev_monitor_new_from_netlink.

Крім того, libudev також містить застарілі функції в API, які не повинні використовуватися новим програмним забезпеченням.

Операційна система Mac OS X надає розробникам можливість використовувати вбодований HID-менеджер для доступу до будь-яких пристроїв, які відповідають специфікації USB HID. Хоча він найчастіше використовується для комунікації з пристроями введення, багато інших типів пристроїв також використовують HID-дескриптори, таким чином, і вони можуть бути доступні за допомогою цього самого менеджеру.

Наприклад, можна використовувати HID-менеджер для отримання інформації з багатьох пристроїв безперебійного живлення (ПБЖ). Пристрої ПБЖ мають однакову структуру дескриптора звітів, як і інші пристрої класу HID, і менеджер надає таку інформацію про них, як напруга, струм і частота. Для керування пристроєм ПБЖ можна отримати доступ до інформації пристрою за допомогою функцій HID-Manager'у і використовувати його для керування процес диспечиризації живленняї.

Інтерфейси HID-менеджеру також можна використовуються як механізм для зчитування та запису невеликих обсягів даних при спілкуванні з певними пристроями, які з якими взаємодіє людина. Наприклад, існують різні загальні інтерфейсні мікросхеми, призначені для забезпечення управління низькою пропускною здатністю і введення з невичислимих пристроїв, таких як контролери двигуна, термістори і так далі.

Менеджер HID для ОС Mac OS X складається з трьох компонентів:

- клієнтський API HID Manager, який надає визначення та функції, які программа розробника ПЗ може використовувати для роботи з пристроями класу HID
- сімейство HID класів, які надають повну інфраструктуру HID в ядрі: базові класи, відображення простору пам'яті ядра користувача, код черги, і парсер HID дескрипторів
- HID драйвери, надані компанією Apple

Розробники клієнського ПЗ, як правило, використовують тільки перший компонент із зазначених у списку клієнтський HID API, який поставляється за допомогою HID-менеджеру.

Менеджер включає такі заголовні файли, як IOHIDLib.h і IOHIDKeys.h (розташовані в /IOKit.framework/Headers/hid), які визначають ключі властивостей, що описують пристрій, ключі елементів, які описують елементи пристрою, і функції інтерфейсу пристрою та структури даних, які використовуються для зв'язку з пристроєм. Після створення інтерфейсу пристрою для вибраного пристрою HID-класу можна використовувати функції інтерфейсу пристрою для відкриття та закриття пристрою, отримання останнього значення елемента або встановлення значення елемента.

Наступний алгоритм коротко описує, як знайти пристрій класу HID, створити для нього інтерфейс пристрою, підключитися до нього і зв'язатися з ним.

1. Пошук об'єкту, який представляє пристрій, в реєстрі вводу-выводу.
2. Створення інтерфейсу для пристрою. Для пристрою HID-класу створюється інтерфейс типу IOHIDDeviceInterface. Цей інтерфейс, визначений в IOHIDLib.h, надає всі функції, необхідні для доступа та взаємодії з пристроєм.

3. Встановлення з'єднання з пристроєм, за допомогою виконання відповідної функції інтерфейсу пристрою.
4. Комунікація з пристроєм, з використанням функцій, які надаються HID-менеджером. Наприклад, щоб отримати останнє значення елемента, використовуйте функцію getElementValue. Функції для створення черги і маніпуляції також надаються HID-менеджером. Наприклад, щоб прочитати наступну подію з черги, необхідно використати функцію getNextEvent; Щоб отримати сповіщення, коли до черги надійде перший елемент, треба використати функцію setEventCallout.
5. Після закінчення роботи з пристроєм, треба викликати відповідну функцію з інтерфейсу для закриття з'єднання.
6. Звільнити інтерфейс пристрою.

3. ОПИС РОЗРОБЛЕНИХ АЛГОРИТМІВ

3.1 Реалізації кросплатформенної бібліотеки для взаємодії з HID пристроями

Для створення HID бібліотеки, яка підтримується трьома різними операційними системами - Windows, Linux та Mac OS X, було створено бібліотеку з власним інтерфейсом. Реалізація інтерфейсу залежить від ОС для якої збирається ця бібліотека. Для кожної з реалізацій використовуються стандартні засоби для взаємодії з USB HID пристроями, які надаються операційною системою для того, щоб уникнути додаткових залежностей.

Для збірки даного роду бібліотеки зручно використовувати кросплатформенну систему автоматичного збирання програмного забезпечення написаного мовою C++ - CMake. Вона дозволяє застосовувати при збірці бібліотеки ті файли, які містять реалізацію інтерфесу для тієї операційної системи, для якої вона вказана в конфігураційному файлі CMakeLists.txt.

Імплементація інтерфейсу для операційної системи Windows використовує HID API, яке надається разом з інструментами WDK.

Під час перерахування підключених пристроїв програмне забезпечення хост-системи визначає інтерфейс підключеного USB пристрою. Після отримання дескрипторів кінцевих точок система опрацьовує кожну кінцеву точку, налаштовану як на переривання IN, так і на переривання OUT на інтервалі, що визначений в дескрипторі кінцевої точки.

Щоб отримати звіти про кінцеві точки, передані пристроєм через канал переривання після опитування від хоста, програма викликає функцію Windows API, що називається Readfile(). Ця функція вимагає параметрів для обробника пристрою, буфера для зберігання інформації, кількості запитаних байтів і змінної, де буде збережено послідовність

успішно отриманих байтів.

Для передачі звіту OUT Endpoint через канал переривання, програма викликає функцію Windiws API, названу Writefile(), і передає параметри цієї функції - обробник пристрою, буфер, що містить звіт, який потрібно передавати, кількість байтів, які потрібно передавати і змінна, де зберігається кількість успішно переданих байтів.

До тих пір, поки пристрій не має даних для передачі через кінцеву точку IN, він повинен просто ігнорувати запити хоста на запит даних, не сигналізуючи, що дані готові до отримання. Після того, як дані були зібрані в структуру звіту і поміщені в буфер IN кінцевої точки, мікропрограми сигналів, що дані готові до передачі.

Коли хост посилає пакет через кінцеву точку OUT, мікроконтролер сигналізує прошивці пристрою, а обробник OUT Endpoint отримує байти з буфера OUT endpoint.

3.2 Алгоритм моніторингу підключених HID пристроїв

Відслідковування подій підключення та відключення пристроїв необхідно для коректної роботи системи взаємодії з пристроями.

Сформовано наступний алгоритм для моніторингу подій підключення та відключення пристроїв:

1. Ініціалізація USB HID бібліотеки;
2. Пошук всіх підключених HID пристроїв;
3. Якщо хоча б один пристрій було знайдено, то перехід до п.5, інакше перехід до п.7;
4. Додання до списку підключених пристроїв;
5. Перехід до п.4;
6. Чи було підключено або відключено HID-пристрій? Якщо так, то перехід до п.8, інакше перехід до п.7;
7. Очищення списку підключених пристроїв

8. Перехід до п.2;

4. АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Особливості реалізації системи та результати

Реалізована система вирішує проблему підтримки сучасним програмним забезпеченням пристроїв взаємодії з користувачем та їх управління.

Система вміє реєструвати певні події від пристроїв, зчитуючи дані з дескрипторів, та реагувати на них. Для захоплення подій було встановлено частоту опитування в 100 Гц, тобто оновлення стану пристрою відбуватиметься кожні 10 мс. Цієї частоти цілком достатньо для того, щоб встигнути зреагувати на всі дії користувача: натискання та відпускання клавіш, пересування курсору мишки тощо.

До основних можливостей реалізованої системи управління користувацькими пристроями було додано можливість створювати власні події, які викликаються у відповідь на натискання вказаних користувачем клавіш.

Перелік реалізованих подій включає:

- Відтворення натискання заданої клавіші
- Вставка заданого тесту
- Зміна яскравості всіх діодів пристрою

Відтворення натискання клавіші є найпростішою реалізованою подією. Для її роботи відбувається моніторинг зміни стану вказаної користувачем клавіші та при першому надходженні нового значення стану, відбувається симуляція натискання іншою клавіші, яка також визначається користувачем меню подій в графічному інтерфейсі програми.

Подія вставки заданого тексту, працює аналогічно попередній описаній події, тільки замість відтворення натискання клавіші, відбувається вставка заданого користувачем тесту.

Останньою з реалізованих подій є зміна яскравості діодів пристрою. Для роботи цієї події недостатньо простого зчитування значень стану клавіш пристрою. Для того, щоб змінити яскравість діодів необхідно вже відправити на сам пристрій відповідну команду, зі значенням яскравості, яку вказав користувач. Прикладом, застосування даної функціональності може бути створення події задання максимальної яскравості світлодіодів клавіатури при натисканні однієї з клавіш та ще однієї події для задання мінімального значення, тобто відключення світлодіодів.

Іншою реалізованою особливістю системи управління є можливість створення ефектів, які відтворюються за допомогою зміни з кольору світлодіодів простою з плином часу за певним алгортмом, який визначає розробник програмного забезпечення. В даному дипломного проекті реалізовно два алгоритми для відтворення таких світлових ефектів.

Перший ефект відображає градієнтний перехід з плином часу між двома кольорами, які користувач задає в меню налаштування світловим ефектів. Ефект є періодичним і не закінчується після першої ж ітерації зміни кольору. Сам період є константним і завжди визначеним в наштуваннях і дорівнює 10 секундам. Під час запуску анімації визначається часова мітка (timestamp) для того, щоб мати точку відліку для її програвання. Анімація відтворюється покадрово зі швидкістю 25 FPS (з англ. frames per second - кадрів в секунду). Для кожного кадру розрховується колір з врахуванням мітки часу початку, періоду та поточній позиції в періоді.

Для того, щоб знайти період (t_{st}) часу, який ефект вже встиг відіграти від початку моменту його початку використовується наступна формула:

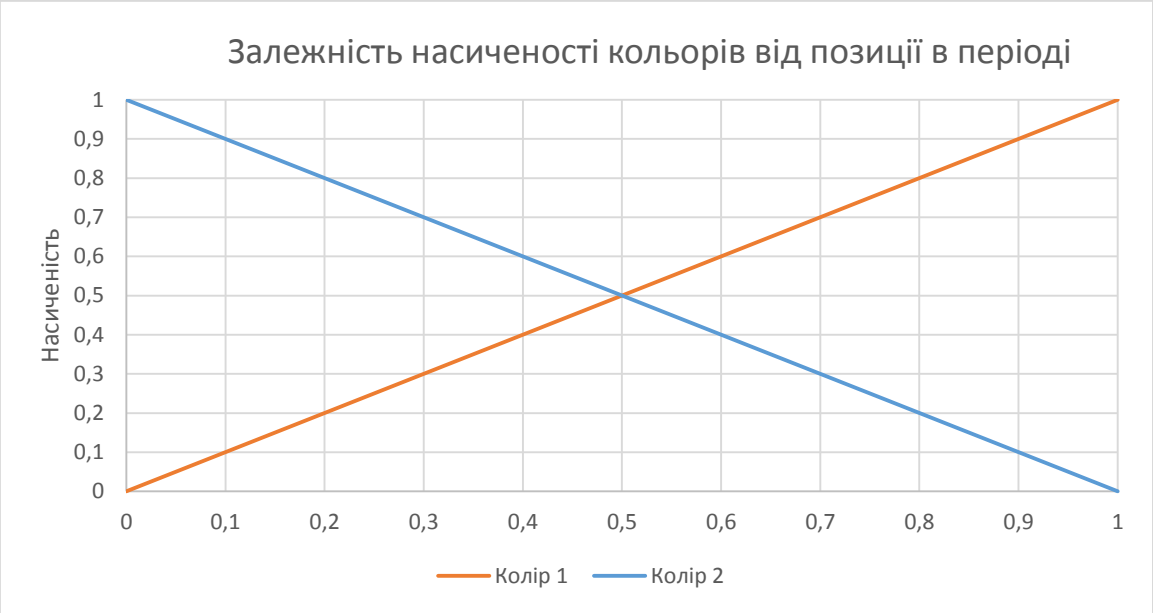
$$t_{st} = t_0 - t$$

Де t – часова мітка в момент розрахунка кольору для кадру

Поточна позиція в періоді (p) розраховується за формулою:

$$p = \frac{t_{st}}{T}$$

Залежність кольору кадру від позиції в періоді для першого алгоритму проілюстрована на графіку 4.1.

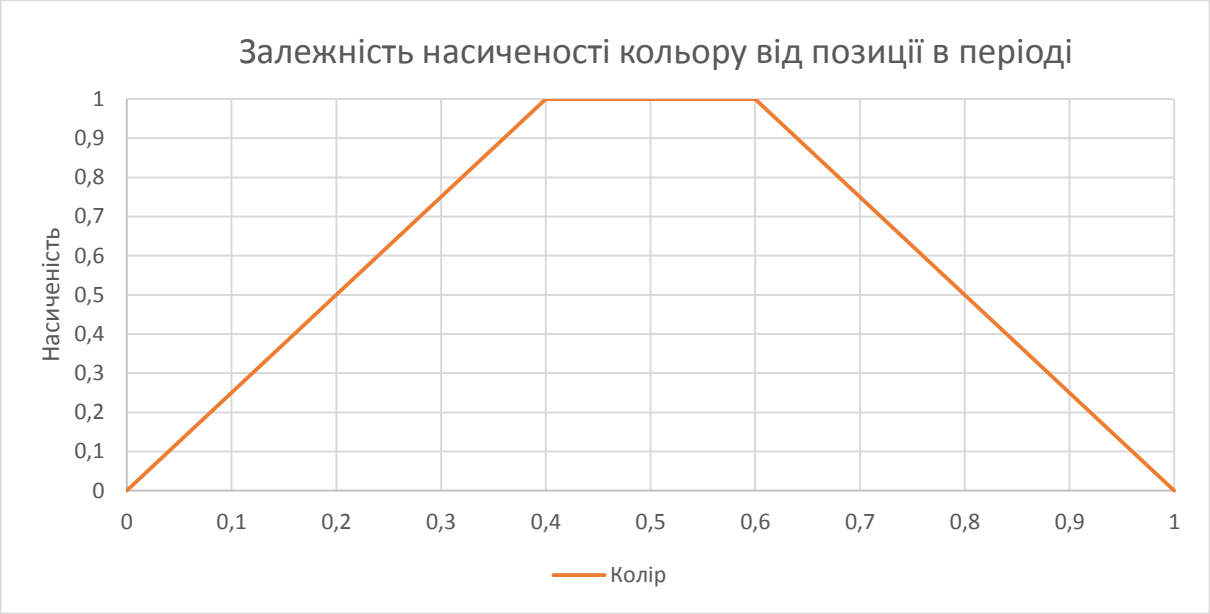


Графік 4.1 – Залежність насиченості кольорів від позиції в періоді

Друга анімація використовує лише один колір, визначений користувачем в меню налаштування ефектів. Він так само як і перший ефект є періодичним і його результуючий колір фрейму залежний від поточної позиції в періоді. Анімація другого ефекту починається з максимально насиченого кольору, що поступово спадає до повністю прозорого (чорного) і знову поступово повертається до максимально насиченого. Виникає так званий ефект затухання та спалаху.

Значення періоду таке ж саме як і для першого ефекту і дорівнює 10 секундам. Час який вже пройшов від початку та позиція в періоді розраховуються аналогічно як і для першого ефету.

Відмінними є лише переходи кольорів, що продемонстровані на графіку 4.2.



Графік 4.2 – Залежність насиченості кольорів від позиції в періоді

Останньою з реалізованих функцій є перегляд вмісту всіх дескрипторів пристроїв. Дескриптори містять всю основну інформацію про USB пристрій, яку може використати розробник ПЗ для створення застосунку для того, щоб воно мало змогу взаємодіяти з користувацькою периферією.

4.2 Тестування системи

В ході тестування системи було перевірено роботу всіх її основних складових: відображення актуального списку підключений пристроїв, можливість реєструвати події, які виконують після зміни одного з станів пристрою, котрий вибирає користувач, створення анімаційних ефектів за допомогою світлодіодів, що розташовані на пристрої, якщо такі присутні в його конфігурації.

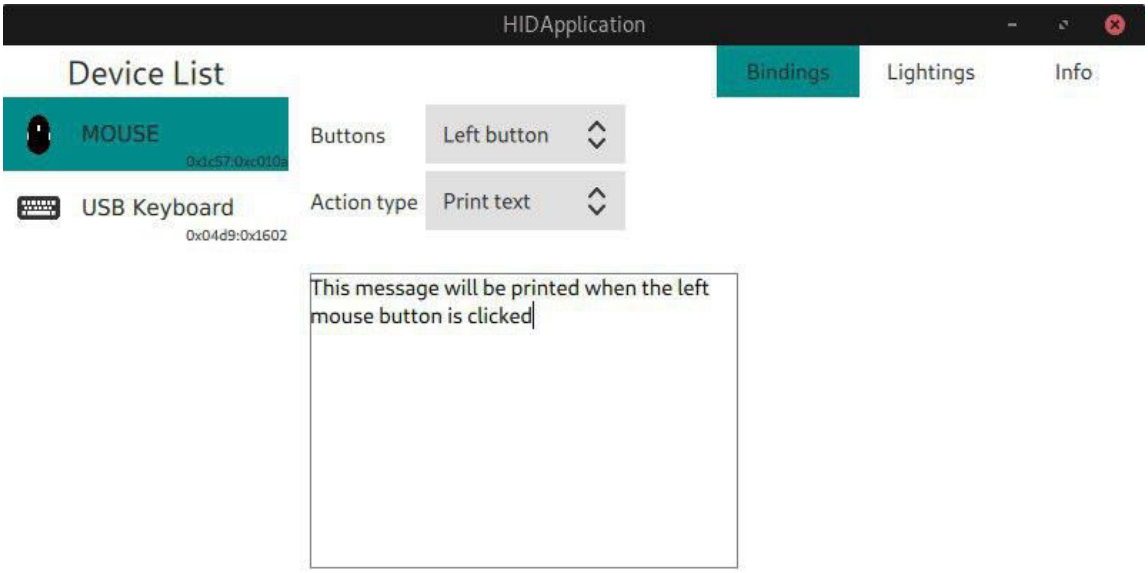


Рисунок 4.1 – Приклад налаштування подій для миші

На рис. 4.1 зображено приклад конфігурації події друкування заданого користувачем тексту після натискання лівої клавіші миші. Вміст тексту може бути довільним і будь-якого розміру. Після успішної конфігурації, задана подія буде працювати в будь-якому вікні, доки працює система управління або доки користувач не змінить конфігурацію.

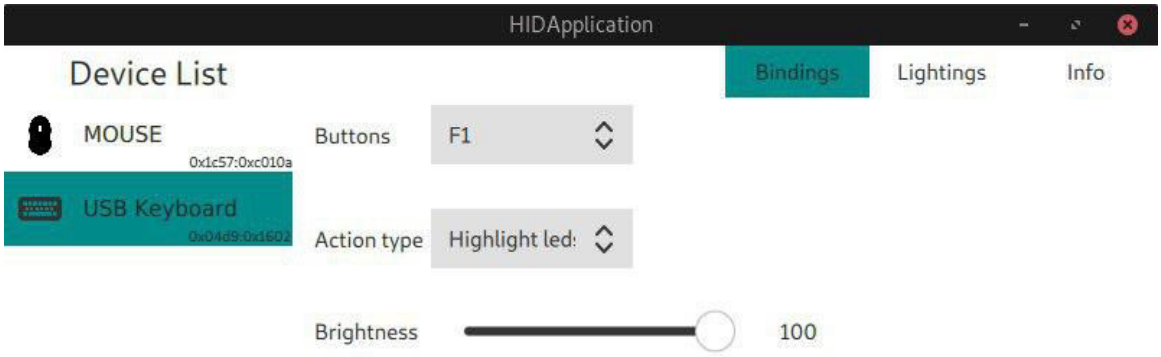


Рисунок 4.2 – Приклад налаштування подій для клавіатури

На рис. 4.2 зображено приклад конфігурації події зміни рівня яскравості світлодіодів клавіатури. Значення рівня освітлення можуть бути задані за допомогою слайдера з допустимими значеннями від 0 до 100 відсотків. В даному випадку значення освітлення дорівнює 100%, тому світлодіоди перейдуть в режим максимального освітлення після того, як користувач натисне на кнопку F1.

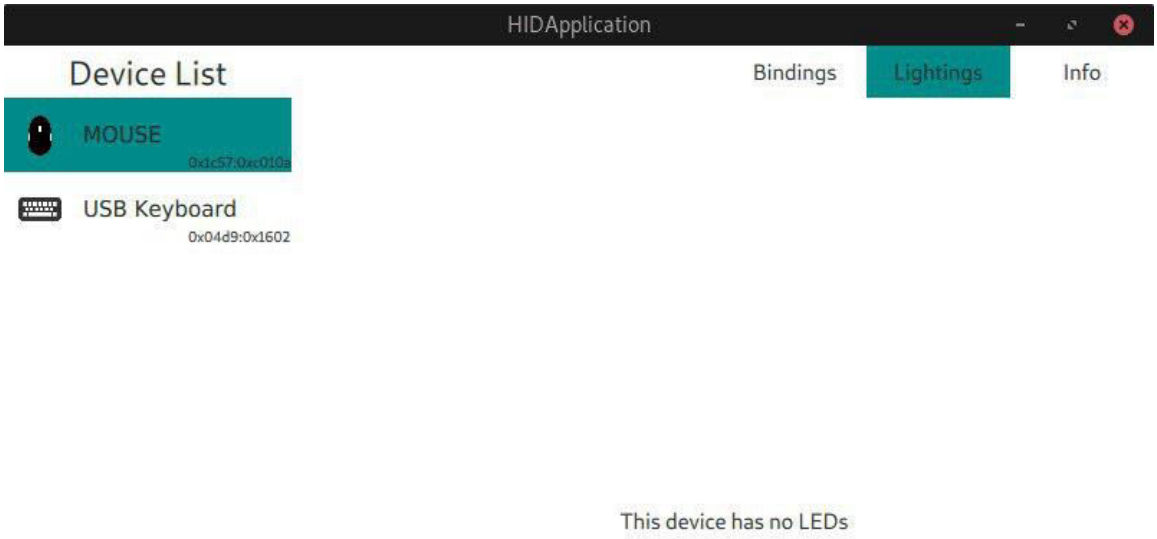


Рисунок 4.3 – Приклад налаштування світлових ефектів для мишки без світлодіодів

На рис. 4.3 зображено приклад налаштування світлового ефекту для мишки, яка не має програмованих світлодіодів в своїй конфігурації. Оскільки вони відсутні, світловий ефект не може бути налаштований та відповідні дані не будуть передані на пристрій. Така поведінка вважається необхідною для уникнення можливих помилок при передачі некоректних даних на пристрій.

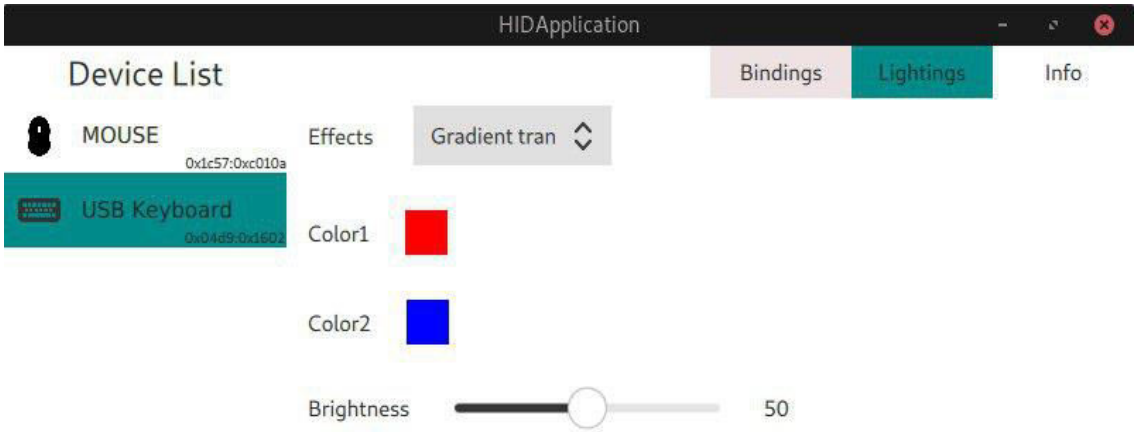


Рисунок 4.4 – Приклад налаштування світлових ефектів для клавіатури

На рис. 4.4 зображено приклад налаштування світлового ефекту для клавіатури з світлодіодами. Для прикладу було обрано ефект з анімацією градієнтного переходу від червоного кольору до синього. Значення рівня максимальної яскравості світлодіодів при цьому встановлено на 50%.

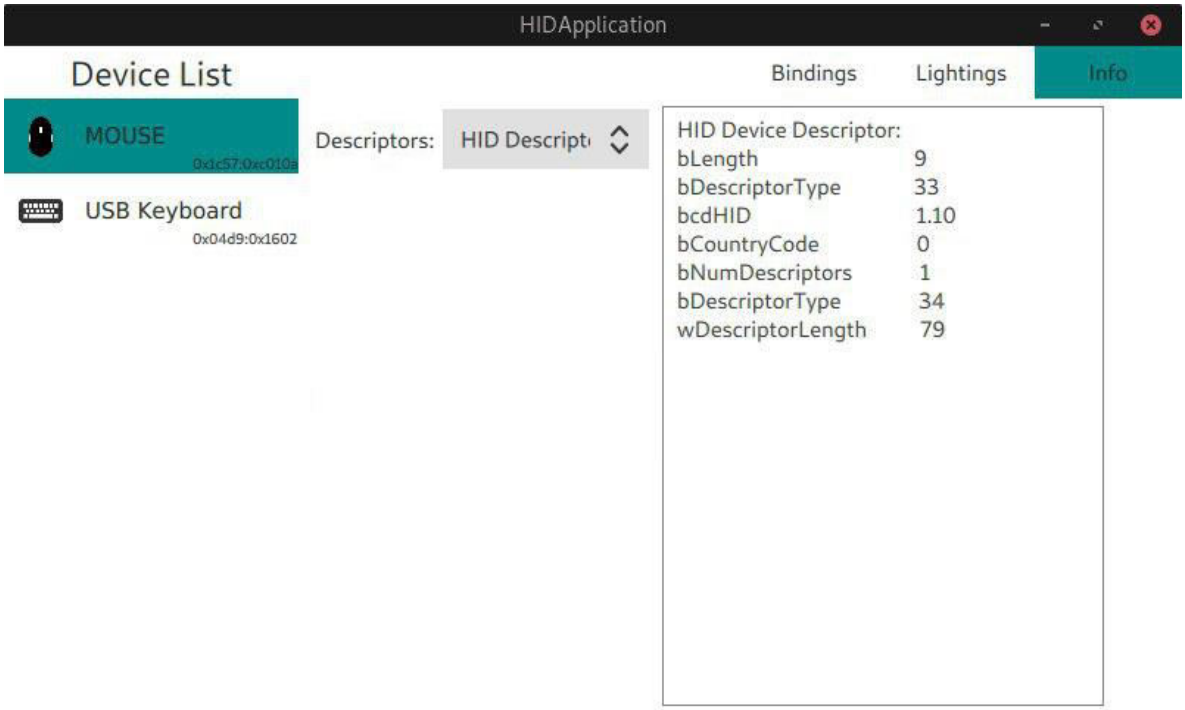


Рисунок 4.5 – Приклад виведення значень полів HID дескриптора для миші

На рис. 4.5 зображено приклад виведення числових значень основних полів HID дескриптора для користувацької миші:

- розмір дескриптора в байтах
- тип HID дескриптора
- номер релізу
- країна для локалізації
- кількість HID дескрипторів
- тип дескриптору звіту
- загальний розмір дескриптора звіту

ВИСНОВКИ

В дипломному проекті було розглянуто проблему розробки системи управління програмованими пристроями взаємодії з користувачем. В процесі розробки було розглянуло наступні питання:

1. Основні інструменти взаємодії з користувацькими пристроями. Дескриптори пристрою, структура звітів та протоколи комунікації.
2. Особливості та переваги використання HID протоколу для взаємодії з пристроями, що підключені через USB шину до персонального комп’ютера.
3. Особливості роботи та аналіз стандартних бібліотек взаємодії з периферією в середовищах найпопулярніших операційних систем – Windows, Linux, MacOS.
4. Створення кросплатформенної бібліотеки для взаємодії з користувацькими пристроями, що працюють за допомогою USB HID протоколу.
5. Реалізація простого монітору подій підключення та відключення HID пристроїв.
6. Створення кросплатформенного додатку з графічним інтерфесом з використанням бібліотеки для взаємодії з користувацькими пристроями.
7. Реалізація основних засобів управління, таких як захоплення подій зміни статусу пристрою, управління світлодіодами та вивід основної інформації про пристрій.

Варто зазначити, що розроблена система управління може бути легко розширена, шляхом створення нових типів подій, що можуть бути корисними під час роботи з персональним комп’ютером та доповненням списку світлових ефектів.

Серед вирішених проблем можна також виділити той факт, серед аналогів системи управління розглянутої в даному дипломному проекті зустрічаються виключно пропрієтарне програмне забезпечення з закритим програмним кодом, яке створюється виробниками дорогої периферії, як правило ігрової, та яке працює виключно з їхньою продукцією.

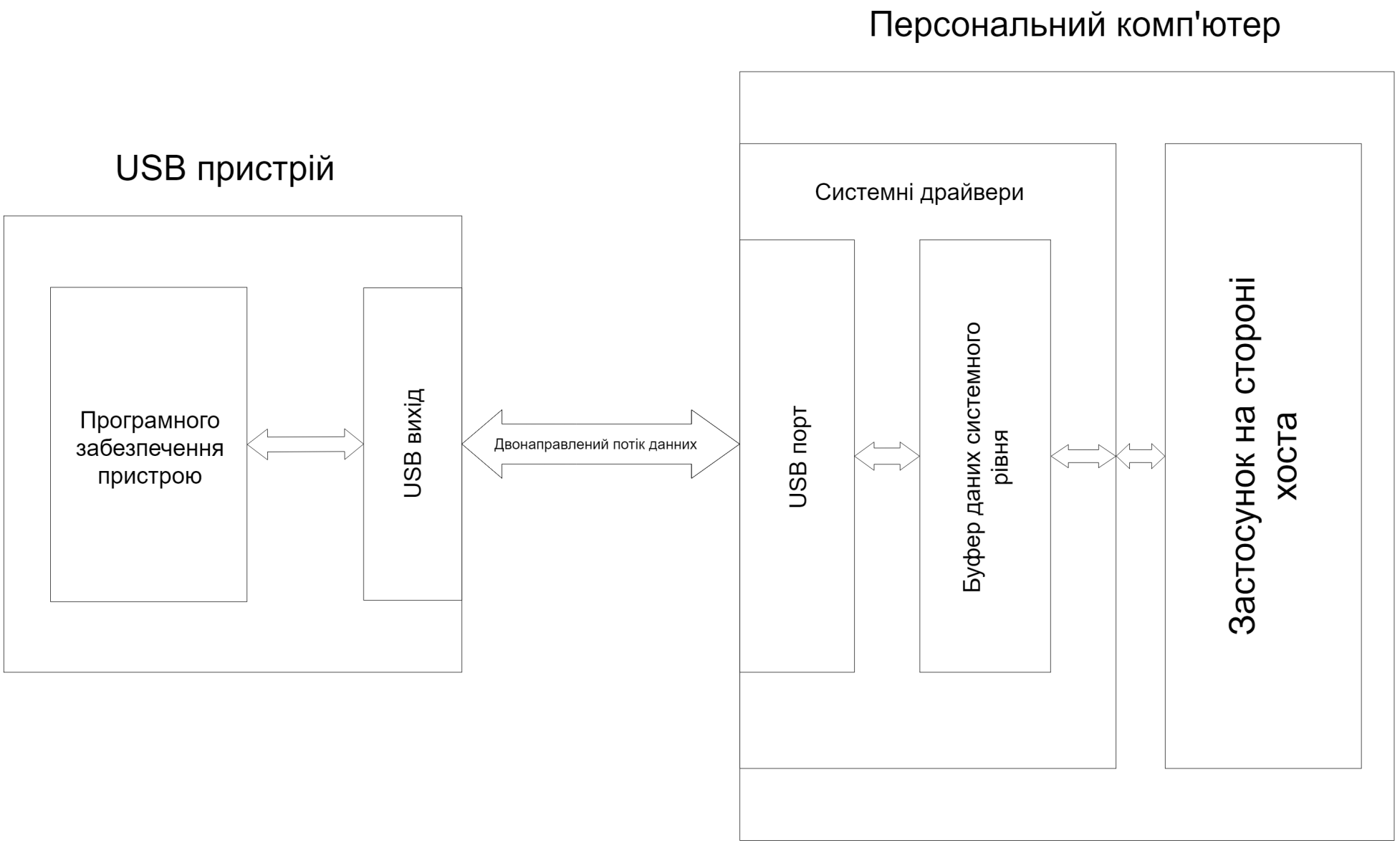
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ларионов А., Горнец Н. Периферийные устройства в вычислительных системах, Учеб. пособие для вузов. М.: Высшая шк., 1991. [Текст] / А. Ларионов, Горнец Н. 336 с.
2. Цилькер Б., Орлов С. Организация ЭВМ и систем: Учебник для вузов. [Текст] / Б. Цилькер, С. Орлов СПб. Пітер, 2004. - 668 с.
3. Гинзбург А., Милчев М., Солоницын Ю. Периферийные устройства. [Текст] / А. Гинзбург, М. Милчев, Солоницин Ю. СПб.: Питер, 2001. – 448 с.
4. Чекунов, Д. Практикум программиста USB-устройств. Часть 1. EZ-USB FX2LP – универсальное USB-решение. //СОВРЕМЕННАЯ ЭЛЕКТРОНИКА. [Текст] / Д. Чекунов 2005 г, № 4, с. 70-77.
5. Чекунов, Д. Практикум программиста USB-устройств. Часть 2. Разработка аппаратно-программного ядра USB-устройства. //СОВРЕМЕННАЯ ЭЛЕКТРОНИКА [Текст] / Д. Чекунов 2005 г, № 5, с. 66-73.
6. Чекунов, Д. Практикум программиста USB-устройств. Часть 2. Разработка аппаратно-программного ядра USB-устройства. //СОВРЕМЕННАЯ ЭЛЕКТРОНИКА. [Текст] / Д. Чекунов 2005 г, № 6, с. 66-74.
7. Чекунов, Д. Практикум программиста USB-устройств. Часть 3. Расширение функций ядра USB-устройства. //СОВРЕМЕННАЯ ЭЛЕКТРОНИКА. [Текст] / Д. Чекунов 2006 г, № 2, с. 70-77.
8. Партыка Т., Попов И. Периферийные устройства вычислительной техники: учеб. пособие. [Текст] / Т. Партика, І. Попов М.: ФОРУМ: ИНФРА-М, 2007. – 432 с.

9. Авдеев, В. Периферийные устройства: интерфейсы, схемотехника, программирование. М.: ДМК Пресс, 2009. . [Текст] / В. Авдєєв 848 с
10. Агуров П. Интерфейсы USB. Практика использования и программирования. – СПб.: БХВ-Петербург, 2004. [Текст] / П. Агуров 576 с.
11. Гинзбург А., Милчев М., Солоницин Ю. Периферийные устройства [Текст] / А. Гинзбург, М. Милчев, Ю. Солоницин СПб.: Пітер, 2001. – 448
12. Павлов, В. Концепция преподавания дисциплин цикла "Система ввода-вывода ПК" (специальность 220100 "Вычислительные машины, комплексы, системы и сети") [Текст] / В. Павлов Вестник Саровского Физтеха.
13. Авдеев, В. Периферийные устройства: интерфейсы, схемотехника, программирование. [Текст] / В. Авдеев М.: ДМК Пресс, 2009. – 848
14. Гук, М. Шины PCI, USB и FireWire. Энциклопедия. [Текст] / М. Гук СПб.: Пітер, 2005. - 540 с.
15. Гук, М. Аппаратные средства PC. Энциклопедия , 2-е изд. . [Текст] / М. Гук СПб: Пітер, 2001. - 928 с.
16. Гук М. Аппаратные интерфейсы ПК. Энциклопедия [Текст] / СПб: Пітер, 2002. - 528 с.
17. Хаммел Р. Последовательная передача данных: Руководство для программиста: Пер. с англ. [Текст] / Р. Хаммел, М.: Мир, 1996. - 752 с
18. Ан П. Сопряжение ПК с внешними устройствами 2-е изд. [Текст] / П. Ан, 2004. – 320 с.
19. Несвижеский В. Программирование аппаратных средств в Windows. [Текст] / В. Несвижеский, 2004. - 880 с.

20. USB in a NutShell Making sense of the USB standard [Електронний ресурс].– Режим доступу: http://www.beyondlogic.org/usbnutshell/usb1.shtml).
21. USB. Транзакции и пакеты [Електронний ресурс]. – Режим доступу: http://perscom.ru/index.php/usb/99-usb-protocol/633-2012-04-04-17-54- 24.html.
22. HUMAN INTERFACE DEVICE TUTORIAL, Silicon Labs, [Електронний ресурс] – Режим доступу: https://www.silabs.com/documents/public/application-notes/AN249.pdf
23. Qt (software), [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Qt_(software)

Додаток 1
Копії графічних матеріалів



						ІАЛЦ.045490.005 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата	Система управління програмованими пристроями взаємоді з користувачем	Літ	Маса	Масштаб
Розроб		Базильського Л.О.						
Перев		Петрашенко А.В.				Аркуш 1	Аркушів 1	
Т.контр.					Взаємодія зовнішніх пристроїв та ПК за допомогою USB інтерфейсу	КПІ ім. Ігоря Сікорського, ФІМ, КВ-51		
Н.контр.		Клятченко Я.М.						
Затв		Тарасенко В.П.						

					ІАЛЦ.045490.006 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	Система управління програмованими пристроями взаємодії з користувачем	Літ / Маса / Масштаб
Розроб.		Базильського Л.О.				
Перев.		Петрашенко А.В.				Аркуш 1 / Аркушів 1
Т.контр.					Схема інтерфейсу USB HID бібліотеки	КПІ ім. Ігоря Сікорського, ФІПМ, КВ-51
Н.контр.		Клятченко Я.М.				
Затв.		Тарасенко В.П.				

						ІАЛЦ.045490.007 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата	Система управління програмованими пристроями взаємодії з користувачем	Літ	Маса / Масштаб
Розроб		Васильський Л.О.					
Перев		Петрашенко А.В.				Аркуш 1	Аркушів 1
Т.контр					Блок-схема монітору HID пристроїв	КПІ ім. Ігоря Сікорського, ФПМ, КВ-51	
Н.контр		Клітченко Я.М.					
Затв		Тарасенко В.П.					

						ІАЛЦ.045490.008 Д4	
Зм.	Арк.	№ докум.	Підпис	Дата	Система управління програмованими пристроями взаємодії з користувачем	Літ	Маса / Масштаб
Розроб		Васильський Л.О.					
Перев		Петрашенко А.В.				Аркуш 1	Аркушів 1
Т.контр.					Структура кросплатформенної HID бібліотеки	КПІ ім. Ігоря Сікорського, ФІМ, КВ-51	
Н.контр.		Клятченко Я.М.					
Затв.		Тарасенко В.П.					